Decoders

• A **full decoder** with \( n \) inputs has \( 2^n \) outputs. Let inputs be labeled \( I_{n_0}, I_{n_1}, I_{n_2}, \ldots, I_{n_{n-1}} \), and let outputs be labeled \( O_{n_0}, O_{n_1}, \ldots, O_{2n-1} \).

• A full decoder functions as follows: Only one of outputs has value 1 (it is active) while all other outputs have value 0. The only output set to 1 is one labeled with the decimal value equal to the (binary) value on input lines.

• In general, a decoder with \( n \) inputs may have fewer than \( 2^n \) outputs. Sometime those are called **partial decoders**.

3-Input Full Decoder

![3-Input Full Decoder Diagram](image)

**Figure B.3.1**
Multiplexers

- A basic multiplexer has only one output line z. There are two sets of input lines: data lines and select lines.

- Let a number of data lines be N, labeled $d_0, d_1, d_2, \ldots, d_{N-1}$. There are m select lines, labeled $s_0, s_1, \ldots, s_{m-1}$. m is such that any of data lines can be referenced (selected) by a decimal value on select lines. Thus, m has to satisfy the following inequality: $2^{m-1} < N \leq 2^m$.

- A multiplexer functions as follows: Output z has the value of the data input line labeled by a decimal value equal to a (binary) value on select lines.

Simplest Basic Multiplexer

Simplest mux is one with 2 data input lines and 1 select line.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Design

Figure B.3.2
### 3-Data Multiplexer Truth Table

<table>
<thead>
<tr>
<th>d0</th>
<th>d1</th>
<th>d2</th>
<th>s1</th>
<th>s0</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>d</td>
</tr>
</tbody>
</table>

*This input not allowed

### Complex Multiplexer

- Instead N single data lines and one output line as in a basic mux, a complex mux has N sets of data lines and one set of output lines and each set has K lines.
- No changes with select lines.

![Design of mux on left at basic mux level](image)

*Figure B.3.6*

N=2, K=32

---

*This input not allowed*
R-S Latch: Simplest Sequential Circuit

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A nor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

1. Let us start with: $S = 0 \& R = 1 \rightarrow Q_a = 0 \& Q_b = 1$
2. Let us now change $R$ to $0$: $S = 0 \& R = 0 \rightarrow Q_a = 0 \& Q_b = 1 \rightarrow$ no change
3. Let us now change $S$ to $1$: $S = 1 \& R = 0 \rightarrow Q_a = 1 \& Q_b = 0$
4. Let us now change $S$ to $0$: $S = 0 \& R = 0 \rightarrow Q_a = 1 \& Q_b = 0 \rightarrow$ no change

Thus, for steps 2 and 4 inputs are identical while outputs are different, i.e. we have a sequential circuit.

Study: B.7 & B.8

---

R-S Latch Characteristics

R-S latch is a memory element that “remembers” which of two inputs has most recently had value 1:

- Outputs $Q_a = 1 \& Q_b = 0$ indicate that $S$ is currently or was 1 last
- Outputs $Q_a = 0 \& Q_b = 1$ indicate that $R$ is currently or was 1 last
5. Let us consider case: \( S = 1 \) & \( R = 1 \) \( \Rightarrow \) \( Q_a = 0 \) & \( Q_b = 0 \)

\( Q_a = 0 \) & \( Q_b = 0 \) indicate that currently \( S = 1 \) and \( R = 1 \)

6a. Let us now change \( S \) to 0: \( S = 0 \) & \( R = 1 \) \( \Rightarrow \) \( Q_a = 0 \) & \( Q_b = 1 \),

\( \Rightarrow \) Identical to Step 1 on Slide 21

6b. Let us now change \( S \) and \( R \) simultaneously to 0: \( S = 0 \) & \( R = 0 \)

\( \Rightarrow \) Unstable state

Note: Set=Reset=1 at the same time should be avoided
• C input is write enable
• When $C = 1$, a gated R-S latch behaves as an ordinary R-S latch, since $R' = R$ and $S' = S$.
• When $C = 0$, changes in $R$ and $S$ do not influence outputs.
• Note that the case $R' = 1$ & $S' = 1$ is still possible, and then an unstable state can be reached easily. How?

(Gated) D-Latch

• Two inputs:
  – the data value to be stored ($D$)
  – the write enable signal ($C$) indicating when to read & store $D$
• Two outputs:
  – the value of the internal state ($Q$) and it's complement (often unused)
• Note: The case $R' = 1$ & $S' = 1$ is not possible.
**D-Latch Functioning**

- **D-latch functions as follows:**
  - when C=1, D-latch state (and Q-output) is identical to D-input, i.e. any change in the value of D-input is ‘immediately followed by the change of Q-output.
  - When C=0, D-latch state is unchanged and it keeps the value it had at the time when C input changed from 1 to 0.

**D Flip-Flop**

- Two inputs:
  - the data value to be stored (D)
  - the write enable signal (C) indicating when to read & store D
- Two outputs:
  - the value of the internal state (Q) and it's complement (often unused)

D flip-flop inputs and outputs identical to that of D-latch.
**D Flip-Flop Functioning**

- **D-flip-flop functions as follows:**
  - When C changes its value from 1 to 0, i.e. on the falling edge, D-flip flop state (and Q output) gets the value D-input has at that moment,
  - During all other times, D-flip flop state is unchanged and it keeps the value it had at the time of the falling edge of C-input.

**Set-up and Hold Time**

- There is critical period $T_{cr}$ around the falling edge of C during which value on D should not change. $T_{cr}$ is split into two parts, the setup time before the C edge, and the hold time after the C edge.
- **The setup time** is the minimum time the D signal must be stable before the C signal falling edge.
- **The hold time** is the minimum time the D signal must be stable after C signal falling edge.

- Failure to meet these minimum requirements can result in a situation where the output of the flip-flop may not be predicable.
- Note that hold times are usually either 0 or very small and thus not a major concern.
Register File Design

- Register file = set of registers that can be read/written by supplying a register number
- D flip-flop - building block

- Typical register file has:
  - 2 read ports
  - 1 write port
- **Read operation:**
  - supply reg# at input
  - output on Read data line
- **Write operation:**
  - select reg#
  - set write data signal